Paper No. 12

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte RAVI KUMAR ARIMILLI, LEO JAMES CLARK,
JOHN STEVEN DODSON and GUY LYNN GUTHRIE

Appeal No. 2003-0263
Application No. 09/340,074

ON BRIEF

Before BARRETT, RUGGIERO, and LEVY, Administrative Patent Judges.
LEVY, Administrative Patent Judge.

DECISION ON APPEAL

This is a decision on appeal under 35 U.S.C. § 134 from the examiner's final rejection of claims 1-7, 10-18 and 21-27, which are all of the claims pending in this application.

BACKGROUND

Appellants' invention relates to a layered local cache with a lower level cache optimizing allocation mechanism. Specifically, the invention relates (specification, page 2) to a method of accessing memory values (operand data or instructions) used by a processor of a computer system. In particular, the present invention relates to a multi-level cache hierarchy, and ports values directly to, e.g., a rename register, instruction buffer, or translation table of the processor without the need for load queues or reload buffers in high level caches.

An understanding of the invention can be derived from a reading of exemplary claim 1, which is reproduced as follows:

1. A method of operating a multi-level cache of a computer system, comprising the steps of:

monitoring cache activity of an upper level cache and a lower level cache both associated with a processor of the computer system, said monitoring including monitoring cache hits in the upper level cache;

issuing a request from the processor to load a value, wherein the request misses the upper level cache and the lower level cache; and

selecting a victim cache block in the lower level cache for receiving the requested value based at least in part on cache hits in the upper level cache.

The prior art references of record relied upon by the examiner in rejecting the appealed claims are:

Patel et al. (Patel)	5,737,751	Apr. 07, 1998

Claims 1-7, 10-18 and 21-27 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Patel.

Rather than reiterate the conflicting viewpoints advanced by the examiner and appellants regarding the above-noted rejection, we make reference to the examiner's answer (Paper No. 7, mailed March 22, 2002) for the examiner's complete reasoning in support of the rejection, and to appellants' brief (Paper No. 6, filed January 15, 2002) and reply brief (Paper No. 9, filed July 30, 2002) for appellants' arguments thereagainst. Only those arguments actually made by appellants have been considered in this decision. Arguments which appellants could have made but chose not to make in the brief have not been considered. See 37 CFR 1.192(a).

OPINION

In reaching our decision in this appeal, we have carefully considered the subject matter on appeal, the rejection advanced by the examiner, and the evidence of anticipation relied upon by the examiner as support for the rejection. We have, likewise, reviewed and taken into consideration, in reaching our decision, appellants' arguments set forth in the briefs along with the examiner's rationale in support of the rejection and arguments in rebuttal set forth in the examiner's answer.

Upon consideration of the record before us, we reverse, essentially for the reasons set forth by appellants. Appellants assert (brief, page 4) that the claims stand or fall together. Consistent with this statement, appellants' remarks are directed to claim 1. Accordingly, we select claim 1 as representative of the group.

To anticipate a claim, a prior art reference must disclose every limitation of the claimed invention, either explicitly or inherently. In re Schreiber, 128 F.3d 1473, 1477, 44 USPQ2d 1429, 1431 (Fed. Cir. 1997). As stated in In re Oelrich, 666 F.2d 578, 581, 212 USPQ 323, 326 (CCPA 1981) (quoting Hansgirg v. Kemmer, 102 F.2d 212, 214, 40 USPQ 665, 667 (CCPA 1939)) (internal citations omitted):

> Inherency, however, may not be established by probabilities or possibilities. The mere fact that a certain thing may result from a given set of circumstances is not sufficient.

The issue before us is whether Patel teaches selecting a victim cache block in the lower level cache for receiving the requested value based at least in part on cache hits in the upper level cache. The examiner's position (answer, page 3) is that Patel selects:

> a victim cache block in the lower level cache for receiving the requested value based at least in part on the prior cache activity of the upper level cache (i.e., the reload queue of the L2 cache is selected for receiving the requested value for sending to both LI cache and L2 cache based on the prior cache misses activity of L1 cache; see column 3 lines 5-10).

In the remarks, the examiner shifts position and asserts (answer, pages 7 and 8) that:

> Patel clearly teaches the selection of a victim cache block by teaching that if there is a memory access miss occurring in accessing on-chip level-one cache (L1), the level-two cache (L2) is then accessed for the requested data (see column 6 lines 59-67 and column 7 lines 1-13). Also, Patel discloses the selection of a victim in a lower level (L2) cache utilizing cache hit information for an upper level cache (L1) by teaching a weak inclusion concept which states that the L1 cache contains a subset of the block of memory in the L2 cache but that changes to the L1 cache are updated in the L2 cache periodically using a write-back operation (see column 2 lines 42-45).

Appellants assert (brief, pages 4 and 5) that col. 3, lines 5-10 of Patel, relied upon by the examiner, teach that when a request of Patel's processor 12 misses both L1 cache 14 and L2 cache 20, the requested data are loaded directly into L1 cache 14, but must be queued in an L2 reload queue within processor 12 prior to transfer to L2 cache 20 via local bus 17. Appellants argue (brief, page 5) that:

> Although the linefill operation disclosed by *Patel* may require selection of a victim cache block in the L2 cache, *Patel* clearly fails to mention selection of the victim cache block and certainly does not teach, suggest or motivate "selecting a victim cache block in the lower level cache ... based at least in part on cache hits in the upper level cache" as claimed.

Appellants add (id.) that:

> Because *Patel* does not disclose the victim cache block selection recited in exemplary Claim 1 explicitly, inherently, or by suggestion, *Patel* fails to render the present claims unpatentable.

Appellants further assert (reply brief, page 2) that:

> In apparent acquiescence to Appellants' position, the Examiner, in paragraph 11 of the Examiner's Answer dated March 22, 2002, fails to respond the Appellants arguments. Instead, the Examiner shifts positions and now relies upon col. 6, lines 59-67, col. 7, lines 1- 13, and col. 2, lines 42-45 of *Patel* as teaching the claimed step and means for selecting a victim cache block in a lower level cache "based at least in part on cache hits in the upper level cache."

It is argued (id.) that:

> Although not expressed in the Examiner's Answer, the implication of the Examiner's reliance upon these two passages is that the L2 cache hits generated by the write-back operations are reflected in a conventional LRU algorithm (*Patel*, Col. 2, lines 14-19) and that the selection of a victim cache block in the L2 cache utilizing the conventional LRU algorithm is therefore "based at least in part on cache hits in the upper level cache" as claimed.

At the outset, we agree with appellants that the examiner has shifted position in the remarks portion of the examiner's answer. In the examiner's rejection, found on pages 3 and 4 of the answer, the examiner's assertion that:

> a victim cache block in the lower level cache for receiving the requested value based at least in part on the prior cache activity of the upper level cache (i.e., the reload queue of the L2 cache is selected for receiving the requested value for sending to both LI cache and L2 cache based on the prior cache misses activity of Ll cache; see column 3 lines 5-10).

refers to sending the requested value to both the Ll and L2 cache based on the prior cache <u>misses</u> activity of the Ll cache. This is not consistent with the language of claim 1 which recites that the victim cache block is selected for receiving the requested value based at least in part on cache <u>hits</u> in the upper level cache. With respect to the examiner's assertion (answer, pages 7 and 8) that:

> Patel clearly teaches the selection of a victim cache block by teaching that if there is a memory access miss occurring in accessing on-chip level-one cache (Ll), the level-two cache (L2) is then accessed for the requested data (see column 6 lines 59-67 and column 7 lines 1-13). Also, Patel discloses the selection of a victim in a lower level (L2) cache utilizing cache hit information for an upper level cache (Ll) by teaching a weak inclusion concept which states that the Ll cache contains a subset of the block of memory in the L2 cache but that changes to the Ll cache are updated in the L2 cache periodically using a write-back operation (see column 2 lines 42-45),

we conclude that the examiner has shifted position from the former position taken in the rejection.

From our review of Patel, we find that Patel is directed to enhanced memory performance through reduced reloads to a second level cache (col. 1, lines 12-14). Patel discloses that when accessing of the disk allows retrieval of necessary data from the cache, such success is called a "hit." When retrieval of necessary data cannot be performed in the cache, such failure is called a "miss" (col. 1, lines 62-65). When a CPU requests a block of data or instructions, and the request misses in the cache, the request is passed on to the main memory (col. 1, line 66 through col. 2, line 1). In order for a new block or cache line to be loaded when the cache is full, blocks must be removed, or castout, from the cache to make room for newly accessed data (col. 2, lines 11-14). A well known and commonly used cache replacement algorithm is a Least Recently Used (LRU) algorithm. According to the algorithm, the block which has not been accessed for the longest period is selected as the least necessary block and is replaced by the new block (col. 2, lines 14-19). Multi-level cache systems divide the cache between an on-chip level-one

(L1) processor cache and a separate discrete level-two (L2) cache to further enhance system performance (col. 2, lines 21-24). The smaller L1 cache is integrated within the processor, and the L2 cache is larger (col. 2, lines 24-26). In the summary of the invention, Patel discloses that when a memory request from the processor causes a miss in both a first level cache and a second level cache, the memory controller loads the retrieved cache line in both the first level cache and the second level cache if the request is a load request, and loads the retrieved cache line in only the first level cache if the request is a store request (col. 3, line 66 through col. 4, line 7). The resultant reduction in reloads to the second cache:

> enhances memory performance by allowing immediate execution of subsequent memory requests to the second level cache and producing a higher hit rate as a result of the reduction in castouts from the second level cache (col. 4, lines 8-12).

From the disclosure of Patel, we find that when upper and lower cache misses occur, the retrieved cache line is loaded into both the upper and lower caches if the request is a load request. We further find that if misses occur in both the upper and lower caches, that the retrieved request is loaded only into the upper cache if the request is a store request. We further find that the selection of the least recently used block for castout to make room for a new block in a cache that is full, is a selection of a victim cache. Shown in figure 2 is a system including a hierarchial memory configuration. Processor 12 includes an on-chip L1 cache 14. L2 cache 20 is connected to the processor by local bus 17 (col. 4, lines 51-54 and 61-63). In operation, cache L2 contains a subset of main memory 22. similarly, L1 cache contains a subset of the block of memory stored in L2 cache 20. Main memory 22 is connected to the system bus 18 through memory controller 24 (col. 4, line 66 through col. 5, line 8). If a memory request is received that causes a miss in both the first level cache and the second level cache, the system passes the address to the main memory to read the datum. According to the embodiment, system 10 enhances memory performance through reduced reloads to the second level cache, upon misses in both the L1 and L2 caches (col. 5, lines col. 5, lines 59-67). Patel further discloses (col. 6, lines 1-17) that the memory request:

> from processor **12** will typically be a store or load request for a cache tine stored in main memory. If the memory request for the cache line causes a miss in both the first level cache and the second level cache, the cache line is accessed in main memory **22** via memory controller **24**. Memory controller **24** retrieves the cache line from main memory **22**. The cache controller function of processor **12** or dedicated cache controller hardware determines if the received memory request is a load request or a store request. If the received memory request is a load request, processor **12** then loads the retrieved cache line in both the first level cache **14** and the second level cache **20**. If the received memory request is a store request, processor **12** loads the retrieved cache line in only the first level cache **14** and not the second level cache **20**. For store instructions, which intend to modify data in the L1 data cache, the linefill needs to be performed only in the L1 cache.

Patel additionally discloses that the system differentiates between load initiated and store-initiated linefills, and that in the prior art, multi-level cache systems are based on strong or weak inclusion, and so require all linefills, whether store initiated or load initiated, to be loaded into both the first and second cache levels. By not forwarding the linefill data to the L2 cache when retrieving a cache line for a store initiated request, the L2 cache reload queues are free for subsequent L1 miss requests (col. 6, lines 19-29). In addition, since the invention does not cause L2 castouts for store-initiated linefills, the hit rate in L2 increases, resulting in better overall memory and processor performance (col. 6, lines 33-36).

From the disclosure of Patel, we find no teaching that the LRU algorithm selects victim cache based at least in part by hits in the upper cache level. We are not persuaded by the examiner's assertion (answer, pages 7 and 8) that:

> Patel clearly teaches the selection of a victim cache block by teaching that if there is a memory access miss occurring in accessing on-chip level-one cache (L1), the level-two cache (L2) is then accessed for the requested data (see column 6 lines 59-67 and column 7 lines 1-13). Also, Patel discloses the selection of a victim in a lower level (L2) cache utilizing cache hit information for an upper level cache (L1) by teaching a weak inclusion concept which states that the L1 cache contains a subset of the block of memory in the L2 cache but that changes to the L1 cache are updated in the L2 cache periodically using a write-back operation (see column 2 lines 42-45).

We find that the portions of Patel relied upon by the examiner relate to weak inclusion (col. 2, lines 42-45) and to a description of a portion of figure 3 (col. 6, line 59-col. 7, line 13), which is consistent with loading the retrieved cache line to both the first and second level cache when the request is a load request, and loading the retrieved cache line in only the first level cache and not to the second level cache if the request is a store request. We agree with appellants (reply brief, pages 2 and 3) that the write-back updates now relied upon by the examiner result in L2 cache hits being reflected in the L2 victim selection i.e., (LRU) algorithm rather than L1 cache hits. In sum, we find no evidence to support the examiner's position that Patel's (LRU) algorithm takes into account cache hit information from the L1 cache. As we stated, supra, inherency cannot be established by possibilities or probabilities. From all of the above, we find that the examiner has failed to establish a prima facie case of anticipation of claim 1. Accordingly, the rejection of claim 1, and claims 2-7 and 10, dependent therefrom, is reversed. As independent claims 11 and 22 also require that the victim cache is selected in the lower cache, based at least in part on cache hits in the upper level cache, the rejection of independent claims 11 and 22, and claims 12-18, 21, and 23-27 dependent therefrom, under 35 U.S.C. § 102(b) is reversed.

CONCLUSION

To summarize, the decision of the examiner to reject claims 1-7, 10-18 and 21-27 under 35 U.S.C. §102(b) is reversed.

REVERSED

[illegible]
LEE E. BARRETT)
Administrative Patent Judge)
)
)
)
[illegible]
JOSEPH F. RUGGIERO) BOARD OF PATENT
Administrative Patent Judge) APPEALS
) AND
) INTERFERENCES
)
[illegible]
STUART S. LEVY)
Administrative Patent Judge)

SSL/vsh

ANDREW J. DILLON
BRACEWELL & PATTERSON LLP
INTELLECTUAL PROPERTY LAW
P.O. BOX 969
AUSTIN, TX 78767-0969